Bureau of Long Term Services and Supports

Utah Electronic Visit Verification (UEVV)

Technical Specifications

Version 1.4

UTAH DEPARTMENT OF
HEALTH
MEDICAID

July 2021

Contact

Please send all inquiries and questions to dmhf_evv@utah.gov.

Table of Contents

1. Introduction

The Utah Electronic Visit Verification (UEVV) project's purpose is to employ automated solutions to achieve compliance with Electronic Visit Verification (EVV) requirements in Section 12006 of the 21st Century CURES Act. This will allow providers to reduce costs associated with EVV and the State to assure claims/encounters associated with Personal Care Services (PCS) and Home Health Services (HHS) can be validated as required by the Act.

The goal is to develop a system for providers to submit EVV records via two transmission methods: an Application Programming Interface (API) channel and a Secure Web portal channel. Both the API and Web portal channels will also support for synchronous status responses (was the transmission accepted or rejected).

1.1. Identification

The purpose of this document is to provide guidance to all types of external transmitters about composing and successfully transmitting compliance EVV data submissions to the State of Utah Medicaid.

The audiences of this document are:

- Provider – A servicing provider who is required to submit EVV records.
- User – A person who works for a servicing provider and is submitting the required data on behalf of the provider.
- Software Developer – the party who is writing either the origination or the transmission software according to UEVV specifications.
- Transmitter – is collectively referred as Provider, User, or Software Developer who is sending the transactions.

1.2. Scope

This document covers details on composing and submitting required EVV records by transmitters to the State of Utah Medicaid. The scope of the document addresses the API-application based via Simple Object Access Protocol (SOAP) messages exchanged between external providers applications and Utah Medicaid's exposed Web Service (WS) endpoints, as well as the Web portal-based channel that allow human initiation to securely submit data.

1.3. Purpose

The purpose of this document is to provide sufficient technical information to the transmitters so that they are able to compose and submit valid data submissions. The document also addresses how the State's acknowledgement response is transmitted to transmitters, as part of the synchronous session as the records are received.

1.4. Definitions, Acronyms and Abbreviations

Table 1: Definitions, Acronyms and Abbreviations

Name	Definition
API	Application Programming Interface
CA	Certificate Authority
CPT	Current Procedural Terminology
CSV	Comma-Separated Value
DSPD	Division of Services for People with Disability
DTS	Department of Technology Services
EVV	Electronic Visit Verification
HCPCS	Healthcare Common Procedure Coding System
HHS	Home Health Services
HTTPS	Hypertext Transfer Protocol Secure
IT	Information Technology
NPI	National Provider Identification
PCS	Personal Care Services
PRISM	Provider Reimbursement Information System for Medicaid
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
SSO	Single Sign-On
UEVV	Utah Electronic Visit Verification
UTC	Coordinated Universal Time
WS	Web Service
WS-Addressing	Web Service Addressing
XML	Extensible Markup Language

2. Interface Overview

The Web interface uses Hypertext Transfer Protocol Secure (HTTPS) for user's data input, submission and receive acknowledgement response. The Utah ID Single Sign-On (SSO) is used to authenticate and authorize user's access to use the portal.

The EVV data is exchanged and encrypted between Transmitters and Utah Medicaid using Secure Socket Layer (SSL) protocol via SOAP message exchange with Extensible Markup Language (XML) file format. The SOAP data structures and XML Schema are specified in this document.

3. Secure Web Portal Channel

In this channel, a user is first presented with a SSO login screen where a Utah ID is required to gain access, if authenticated and authorized, the user will be redirected to the Web interface to initiate and submit EVV data.

3.1. CSV File

The CSV File method is for third-party software solutions to export electronically captured EVV data in a format the Utah Medicaid system will accept. The CSV File method is not to be used as a manual input method.

To submit via CSV, choose the **Upload CSV** from the navigation menu in the upper right.

Download a copy of the CSV template. Use the template to ensure necessary formatting and column order for your CSV to be uploaded to the EVV system. Required fields are marked in the header row with (req) after the column names. If using Excel to adjust the CSV file see Appendix A.

Please note:

1. If using your own CSV file, the first row will always be treated as a header and thus ignored.
2. The upload process may take up to 20 minutes for large files.

You will receive a submission message upon completion of processing. It will tell you how many records were submitted, how many were accepted, and how many were rejected. Below the receipt message you will find information on which rows were rejected. You will need to fix these rows and resubmit them. Please resubmit only the corrected rejections the same way as the initial submission.

The follow table displays the user's options:

Table 3: Upload CSV Options and Functionalities

User Option	Functionality
"Download CSV template" button	Initializes the download process for the CSV template.
"Choose File" button	Opens the file browser and allows a single CSV file to uploaded. After selection the file name is displayed adjacent to the button.
"Upload CSV" button	Initializes the CSV file upload. A status bar will appear and show the progress.

3.2. EVV Data Elements

The following table provides information regarding the required input EVV data elements using the Web portal or CSV.

Table 4: The Input EVV Data Elements

Data Elements	Data Description	Required	Expected Format
Member	Member ID – Medicaid Member ID Member First Name MI Member Last Name	Yes Yes No Yes	VARCHAR (10) VARCHAR (1-255) VARCHAR (0-255) VARCHAR (1-255)
Service	Service Code (HCPCS/CPT code or DSPD service code) Service Description	Yes No	VARCHAR (5) VARCHAR (1-255)
Servicing Provider	Provider NPI or Provider PRISM ID Name of employee performing service	Yes Yes	VARCHAR (1-12) VARCHAR (1-255)
Begin Date of Service	Begin date, Start time	Yes	DATE/TIME
Beginning Service location	Begin Street Address Begin Apt/Suite/Floor Begin City Begin State Begin Zip Begin Geo Latitude Begin Geo Longitude	Yes* No Yes* No No No* No*	VARCHAR (1-255) VARCHAR (0-255) VARCHAR (1-60) VARCHAR (0-20) VARCHAR (0-10) NUMBER (38, 10) NUMBER (38, 10) Example: 28.523
End Date of Service	End date and End time	Yes	DATE/TIME
Ending Service location	End Street Address End Apt/Suite/Floor	No No	VARCHAR (1-255) VARCHAR (0-255)

Data Elements	Data Description	Required	Expected Format
	End City End State End Zip End Geo Latitude End Geo Longitude	No No No No No	VARCHAR (1-60) VARCHAR (0-20) VARCHAR (0-10) NUMBER (38, 10) NUMBER (38, 10) Example: 28.523
Receipt ID	Original Receipt-ID of previous submitted record. **Required if submit a correction record.**	No	VARCHAR (32)
Batch ID	Submitter may select the method to create Batch IDs. **Required.** Original Batch-ID of previous submitted record, if submitting a correction.	Yes	VARCHAR (1-10)
Record ID	Submitter may select the method to create Record IDs. **Required.** Original Record ID of previous submitted record, if submitting a correction.	Yes	VARCHAR (1-10)

NOTE: *Either Begin Address/Begin City OR Begin Geo latitude/Begin Geo longitude must be present.

3.3. Input Data Validations and Error Message

The Web application will check for valid input data fields and the table below provides information regarding the front-end validation and error messages.

Table 5: Input Data Validations and Error Messages:

Input Field	Data Input Validations	Error Message
Member ID	● Not Null	● Member ID invalid. Check for missing leading zeros. ● One or more mandatory fields have not been completed. Please complete the fields and re-submit.

Input Field	Data Input Validations	Error Message
First Name	● Not Null	● One or more mandatory fields have not been completed. Please complete the fields and re-submit.
Last Name	● Not Null	● One or more mandatory fields have not been completed. Please complete the fields and re-submit.
Service Code	● Not Null	● One or more mandatory fields have not been completed. Please complete the fields and re-submit.
Provider ID	● Not Null ● Numeric value	● One or more mandatory fields have not been completed. Please complete the fields and re-submit.
Employee Performing Service	● Not Null	● One or more mandatory fields have not been completed. Please complete the fields and re-submit.
Start Date/ Start Time	● Not Null ● MM/DD/YYYY ● Not greater than End Date/End Time ● Not exceed 365 days ● Cannot be a future date ● Individual service duration cannot exceed 24 hours	● One or more mandatory fields have not been completed. Please complete the fields and re-submit. ● Please enter date as MM/DD/YYYY ● End Service date and time should be greater than Begin Service data and time. ● Service date/time cannot exceed 1 year from Date of Service. ● Service date/time cannot be a future date/time. ● Service duration cannot exceed 24 hours.
End Date/ End Time	● Not Null ● MM/DD/YYYY	● One or more mandatory fields have not been completed. Please complete the fields and re-submit ● Please enter date as MM/DD/YYYY

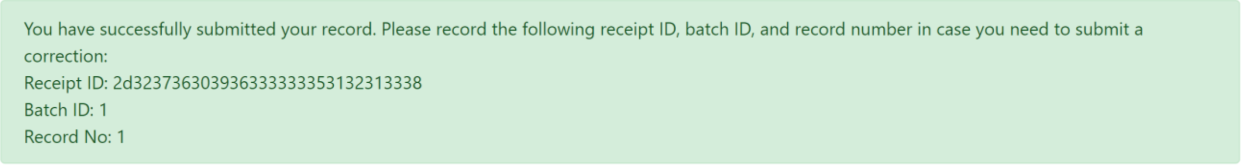
Input Field	Data Input Validations	Error Message
	● Not greater than End Date/End Time ● Not exceed 365 days ● Cannot be a future date ● Individual service duration cannot exceed 24 hours	● End Service date and time should be greater than Begin Service data and time. ● Service date/time cannot exceed 1 year from Date of Service. ● Service date/time cannot be a future date/time. ● Service duration cannot exceed 24 hours.
Begin Street Address	● Not Null (without Begin Geo latitude and Begin Geo longitude) ● Null permitted (with Begin Geo latitude and Begin Geo longitude	● The Street 1 field is empty. *Either Begin Address/Begin City OR Begin Geo latitude/Begin Geo longitude must be present. Please complete the field and re-submit.*
Begin City	● Not Null (without Begin Geo latitude and Begin Geo longitude) ● Null permitted (with Begin Geo latitude and Begin Geo longitude	● The City field is empty. *Either Begin Address/Begin City OR Begin Geo latitude/Begin Geo longitude must be present. Please complete the field and re-submit.*
Begin/End Zip	● Null	● The Zip exceeds the 10-character length.
Batch ID	● Not Null	● Missing Batch ID. ● The Batch ID number field contains invalid characters that are not numbers.
Record ID	● Not Null	● Missing Record ID ● The Record ID number field contains invalid characters that are not numbers.

3.4. Receiving Submission Acknowledgement Response

Note: It is important for the user to capture and save this information for future reference, should they need to submit a correction to replace this record.

Once a record is submitted, the acknowledgement response (including the Receipt ID, batch ID and record ID) displays on the screen for the user's records and future reference.

Figure 1: Web Portal Submission Status and Acknowledgement



You have successfully submitted your record. Please record the following receipt ID, batch ID, and record number in case you need to submit a correction:
Receipt ID: 2d3237363039363333333353132313338
Batch ID: 1
Record No: 1

3.5. Correcting a Previous Entry

To submit a correction to a previous record, you will need to have the Original Receipt ID, Batch ID, and Record ID from the previous submission to populate the CSV template.

In the CSV template, it is important the "Orig_receipt_id" column is formatted as text. Otherwise the system will truncate the entry and the file will fail.

Check the Correction checkbox then upload the CSV file.

4. API Channel

In this channel, records are transmitted using the SOAP Web Services request-response model. An active provider's SSL Certificate must be sent to Utah Medicaid to be installed in the system in advance. Once the certificate is received and stored in the database, a Web Service Description Language (WSDL) endpoint will be communicated to the provider's identified responsible official or contact representative for the provider to set up and start the data submission.

The file submission SOAP messages from the providers are encrypted and will carry the authentication key in file's header for authentication and authorization process. The acknowledgement of file receipt will be returned as a synchronous XML SOAP message to the submitting providers.

A transmission consists of two parts: the Header and the Data File.

- The Header contains information about the transmitter, transmission and the payload
- The data file contains one or more submissions in XML format.

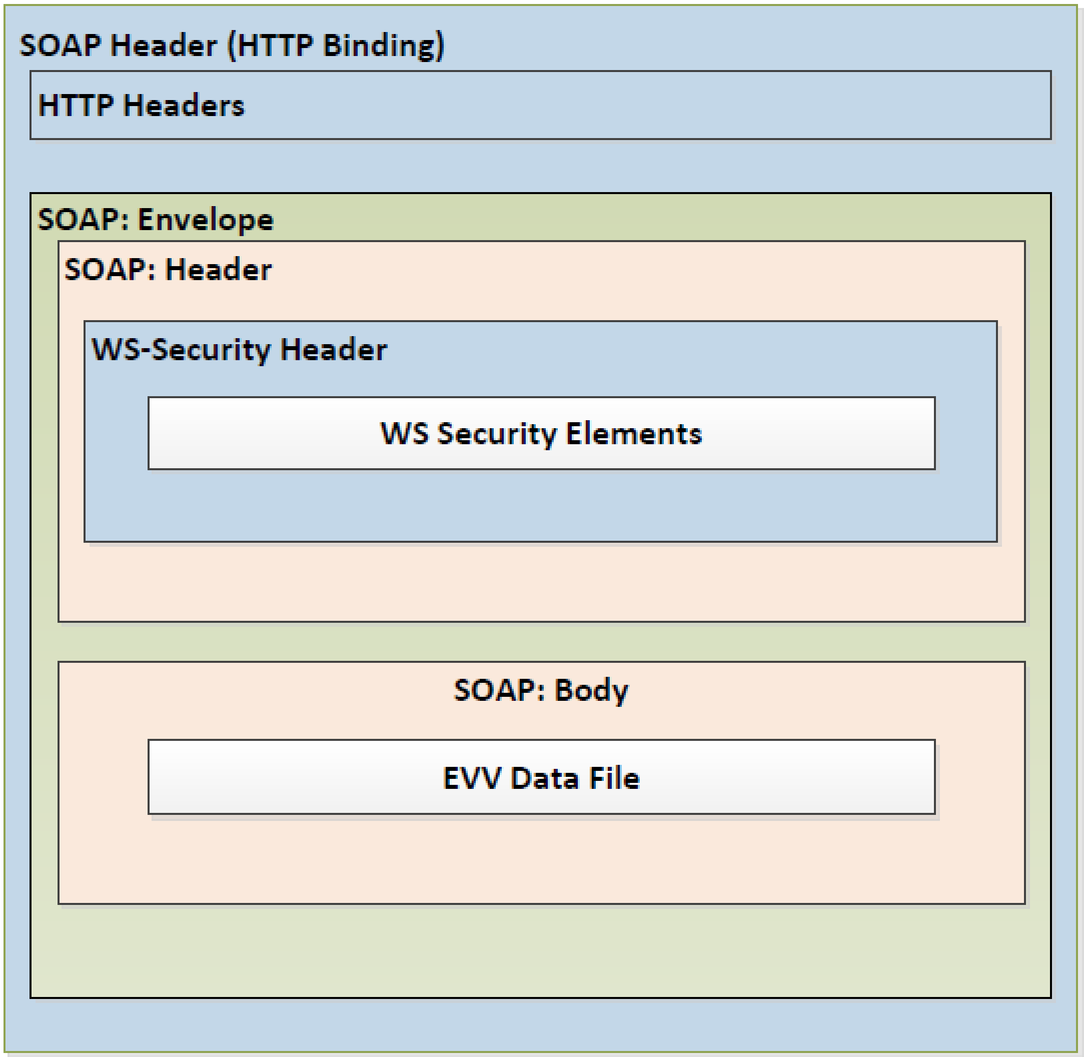
4.1. Basic SOAP Message Structure

A SOAP message is a XML structure consisting of *SOAP Envelope, SOAP Header and SOAP Body* which may contain payload data. A SOAP message starts with an XML declaration <?xml version=”1.0” encoding=”UTF-8”?>. The following section explains various segments and aspects of a SOAP message available to the transmitters. Please be advised that this document is not intended as a tutorial and therefore covers only important aspects of a SOAP message. Please refer to www.w3.org/TR/soap/ and other authoritative websites for more information.

The below figure describes the logical structure of basic messages with a SOAP Header and SOAP Body blocks within a SOAP Message Envelope. A SOAP message contains one SOAP Header and one SOAP Body within one SOAP Envelope.

- The SOAP Header contains the Web Services Addressing (WS-Addressing) and WS-Security,
- The SOAP Body contains the payload structure for the required EVV data file to be submitted.

Figure 2: An Example of SOAP Message Structure:



4.1.1. Creating the SOAP Envelope

The SOAP Envelope consists of a SOAP header and a SOAP body. The SOAP header contains information about the transmitter, the transmission and metadata about the payload in the SOAP body. The SOAP body is also referred to as the content file, EVV Data File or simply payload.

4.1.2. SQL Injection in the SOAP Message

The special characters listed below are treated as Structured Query Language (SQL) injections. SQL injections exploit security vulnerabilities in an application's software, and are mostly known as an attack vector for websites or API communication between two sources. These may allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, or destroy the data. Characters that are not allowed due to concerns about SQL Injections are shown in this table. If needed, the escape characters shown below can be used and are allowed.

Table 6: SQL Injection In The SOAP Message

Character	Character Description	Character Allowed?	Escape Characters	Escape Character Allowed
&	Ampersand	Rejected (malformed check)	&amp	Allowed
‘	Apostrophe	Rejected (sql injection check)	&apos;	Allowed
“	Quotation Mark	Allowed	&quot;	Allowed
--	Double Dash	Rejected (sql injection check)	Not Available	N/A
#	Hash Key	Rejected (sql injection check)	Not Available	N/A
<	Less Then	Rejected (malformed check)	&lt;	Allowed
>	Greater Than	Allowed	&gt;	Allowed

Note: Allowed escaped characters may be identified as a potential threat (Error Code TPE 1204) when they are used in conjunction with certain words such as “and” and “or”, as in “&apos;OR”. If this occurs troubleshoot by removing the apostrophe.

4.2. SOAP Header

Utah Medicaid defines what should be in the SOAP header. The following sections describe the elements in the SOAP header.

4.2.1. WS-Security

Utah Medicaid EVV Web Services comply with Web Services Security (WS-Security) specification version 1.0 for implementing end to end message security. It is an open standard published by OASIS that defines mechanisms for signing and encrypting SOAP messages and provides transport-neutral mechanisms to enforce integrity and confidentiality on messages and allows the communication of various security token formats.

WS-Security defines SOAP extensions to implement client authentication, message integrity and message confidentiality on the message level. Authentication helps identify the Sender (the transmitter). Message integrity ensures the recipient receives unaltered request. XML Signature specification ensures integrity of the message, which defines a methodology for cryptographically signing XML. Message confidentiality is to make sure that the data can't be read during transit. The XML Encryption specification is the basis to encrypt the parts of SOAP message including headers, body blocks, and substructures, which may be encrypted.

To consume Utah Medicaid EVV web services, transmitter must use the X.509 authentication framework with the WS-Security specification. An X.509 certificate specifies a binding between a public key and a set of attributes that includes (at least) a subject name, issuer name, serial number, and validity interval. An X.509 certificate may be used to validate a public key that may be used to verify a SOAP message element or to identify the public key with SOAP message that has been digitally signed.

4.2.1.1. Creating the XML Signature

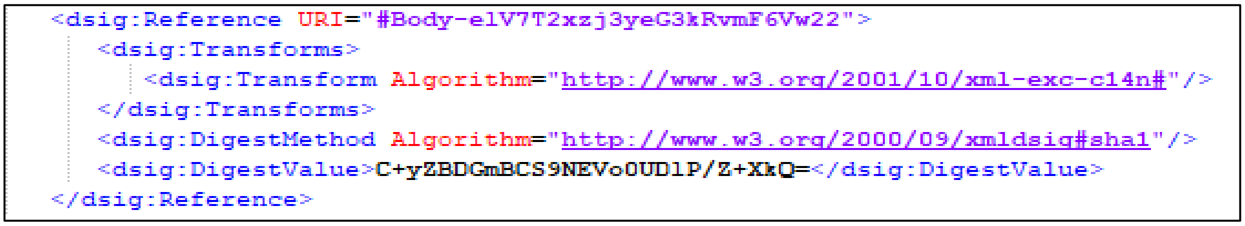
The signatures are defined using a <Signature> element and accompanying sub-elements as part of a security header. Note that the signature must be created after the content of the message is finalized. If changes are made to the message after the signature is created, it may result in a digest mismatch.

Below is a quick overview of how to create an XML signature. Note that XML Digital Signature APIs and XML Digital Signature libraries are also publicly available that may simplify development. An overview and tutorial can be found using the following URL: http://docs.oracle.com/javase/7/docs/technotes/guides/security/xmldsig/XMLDigitalSignature.html

1. Determine which resources are to be signed.

2. Calculate the digest for each resource:

Each referenced resource is specified through a <Reference> element and its digest (calculated on the identified resource and not the <Reference> element itself) is placed in a <DigestValue> child element mentioned in XML snippet below.

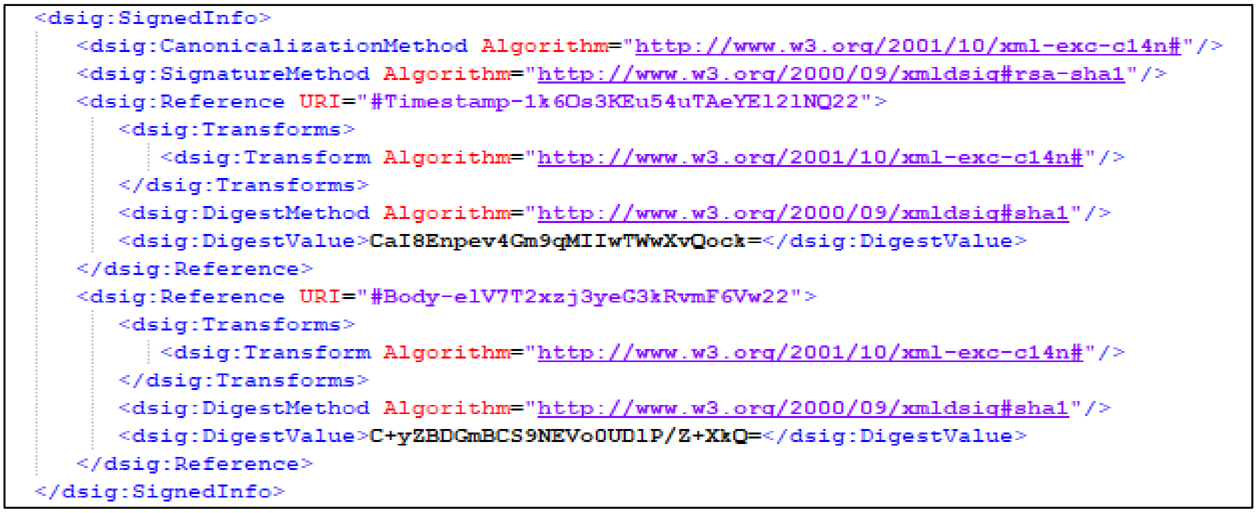


```
<dsig:Reference URI="#Body-elV7T2xzj3yeG3kRvmF6Vw22">
  <dsig:Transforms>
    <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  </dsig:Transforms>
  <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <dsig:DigestValue>C+yZBDGmBCS9NEVo0UD1P/Z+XkQ=</dsig:DigestValue>
</dsig:Reference>
```

The <DigestMethod> element identifies the algorithm used to calculate the digest.

3. Collect the Reference elements:

Collect the <Reference> elements (with their associated digests) within a <SignedInfo> element as shown below. Note that InclusiveNamespaces cannot be a child element of the CanonicalizationMethod element.



```
<dsig:SignedInfo>
   <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
   <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
   <dsig:Reference URI="#Timestamp-1k6Os3KEu54uTAeYE121NQ22">
      <dsig:Transforms>
         <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </dsig:Transforms>
      <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <dsig:DigestValue>CaI8Enpev4Gm9qMIIwTWwXvQock=</dsig:DigestValue>
   </dsig:Reference>
   <dsig:Reference URI="#Body-elV7T2xzj3yeG3kRvmF6Vw22">
      <dsig:Transforms>
         <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </dsig:Transforms>
      <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <dsig:DigestValue>C+yZBDGmBCS9NEVo0UD1P/Z+XkQ=</dsig:DigestValue>
   </dsig:Reference>
</dsig:SignedInfo>
```

The <CanonicalizationMethod> element indicates the algorithm was used to canonize the <SignedInfo> element. Different data streams with the same XML information set may have different textual representations, e.g. differing as to whitespace. The <SignatureMethod> element identifies the algorithm used to produce the signature value.

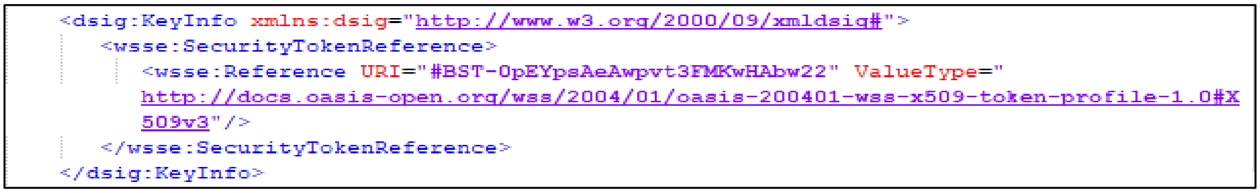
4. Signing:

Calculate the digest of the <SignedInfo> element, sign that digest and put the signature value in a <SignatureValue> element.

```
        <ds:SignatureValue>SignatureValue</ds:SignatureValue>
```

5. Add key information:

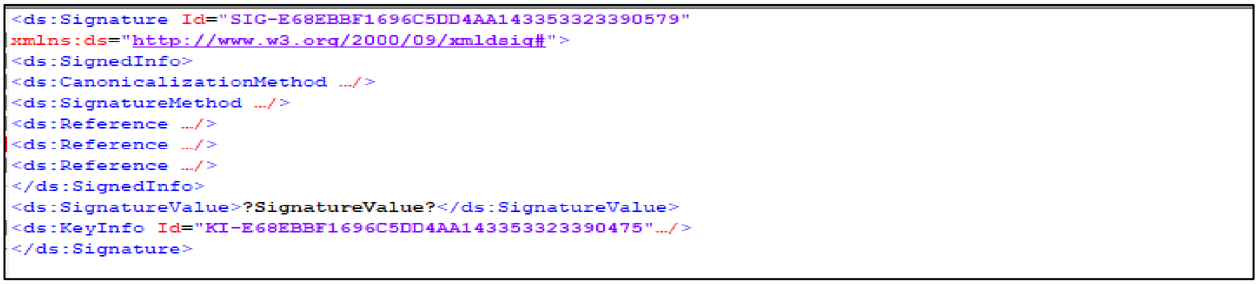
If keying information is to be included, place it in a <KeyInfo> element. Here the keying information contains the X.509 certificate for the transmitter, which would include the public key needed for signature verification.



```
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
   <wsse:SecurityTokenReference>
      <wsse:Reference URI="#BST-0pEYpsAeAwpvt3FMKwHAbw22" ValueType="
      http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X
      509v3"/>
   </wsse:SecurityTokenReference>
</dsig:KeyInfo>
```

6. Enclose in a Signature element

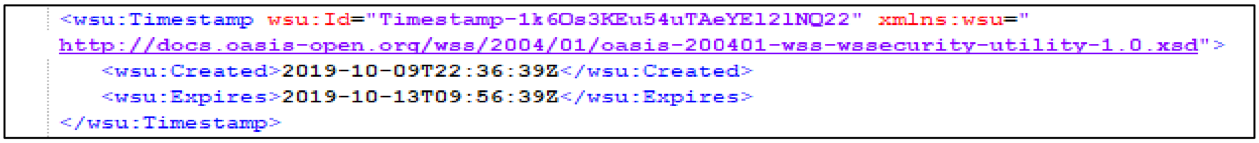
Place the <SignedInfo>, <SignatureValue>, and <KeyInfo> elements into a <Signature> element. The <Signature> element comprises the XML signature.



```
<ds:Signature Id="SIG-E68EBBF1696C5DD4AA143353323390579"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod …/>
<ds:SignatureMethod …/>
<ds:Reference …/>
<ds:Reference …/>
<ds:Reference …/>
</ds:SignedInfo>
<ds:SignatureValue>?SignatureValue?</ds:SignatureValue>
<ds:KeyInfo Id="KI-E68EBBF1696C5DD4AA143353323390475"…/>
</ds:Signature>
```

4.2.1.2. Message Timeout

WS-Security provides the <Timestamp> header can be used to record creation and expiration time of a request message. Note that the WS Timestamp must be used within 30 minutes of creation time based on Coordinated Universal Time (UTC).

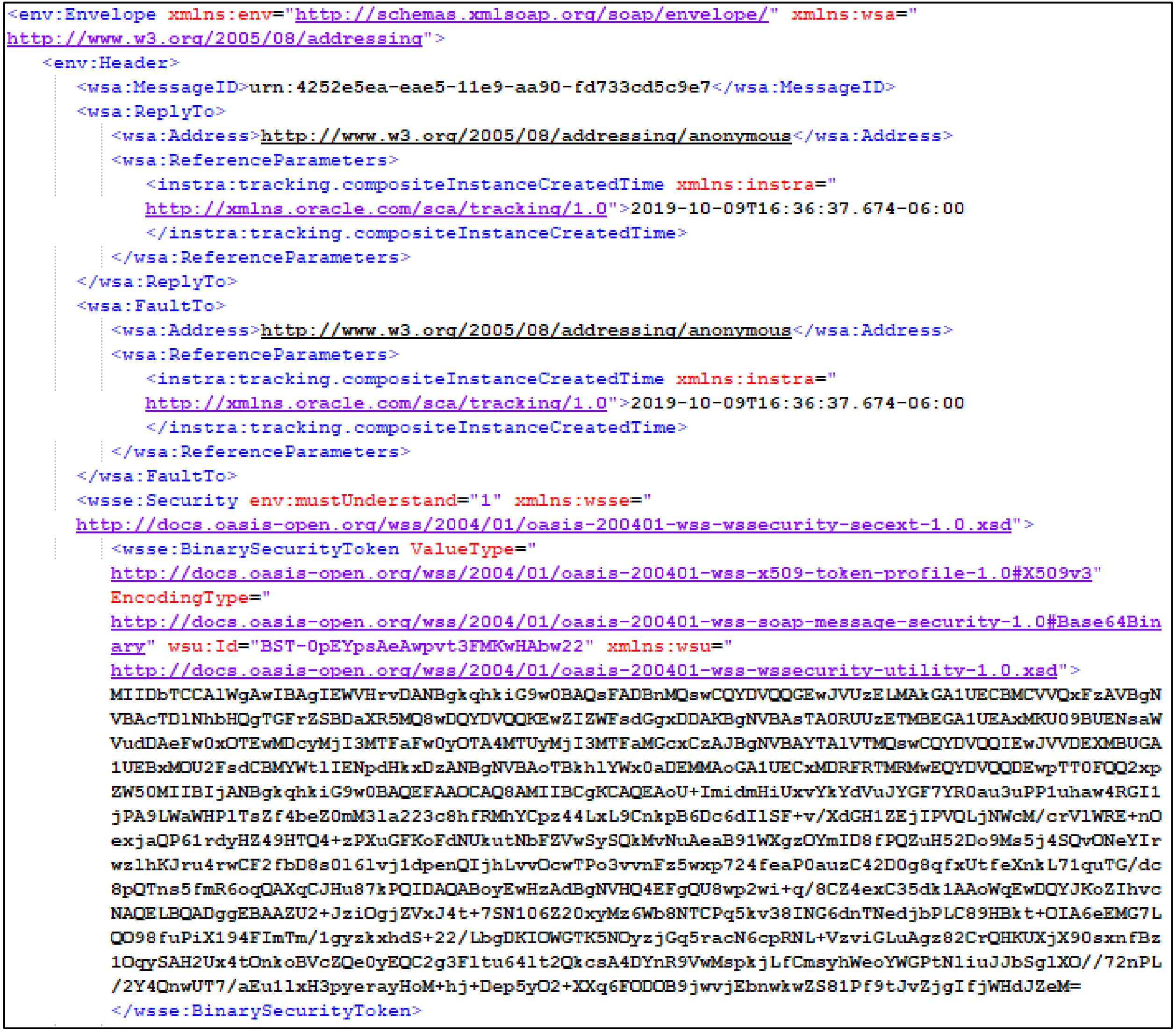


```
<wsu:Timestamp wsu:Id="Timestamp-1k6Os3KEu54uTAeYE121NQ22" xmlns:wsu="
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <wsu:Created>2019-10-09T22:36:39Z</wsu:Created>
   <wsu:Expires>2019-10-13T09:56:39Z</wsu:Expires>
</wsu:Timestamp>
```

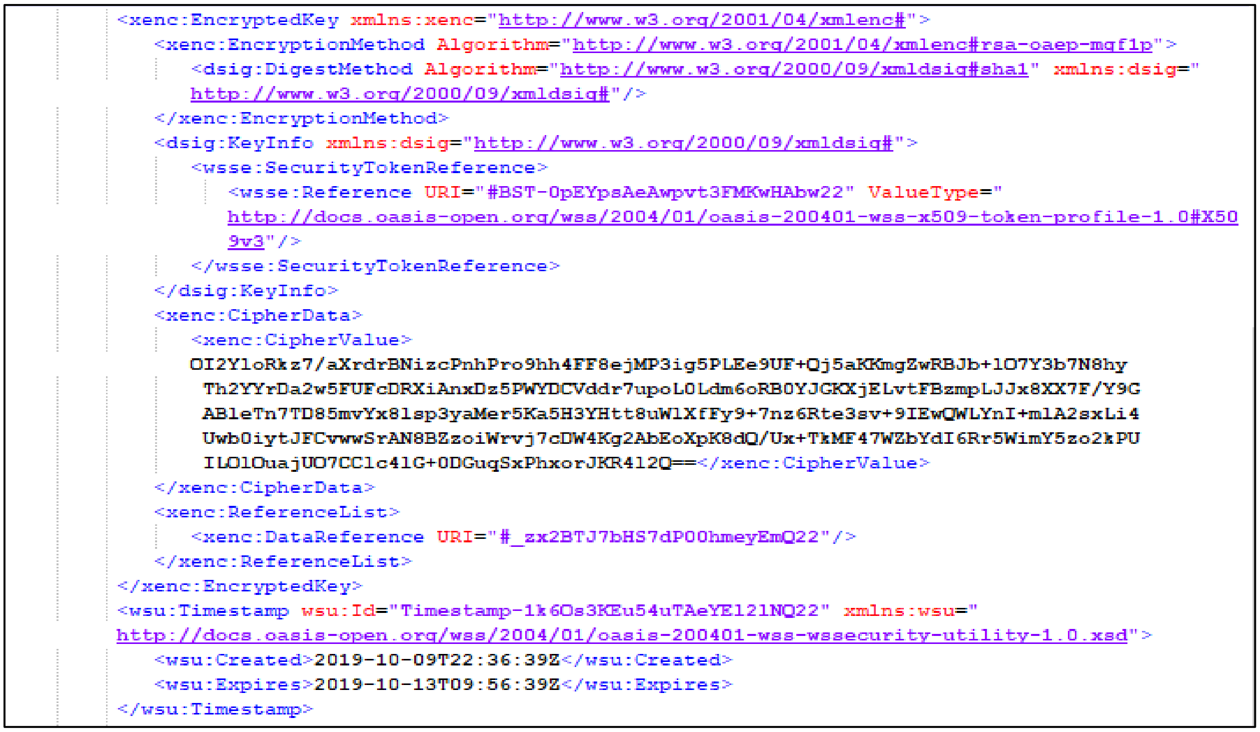
4.2.1.3. SOAP Header Example Showing Security Header and Related Elements

Below is a sample SOAP message transmission.

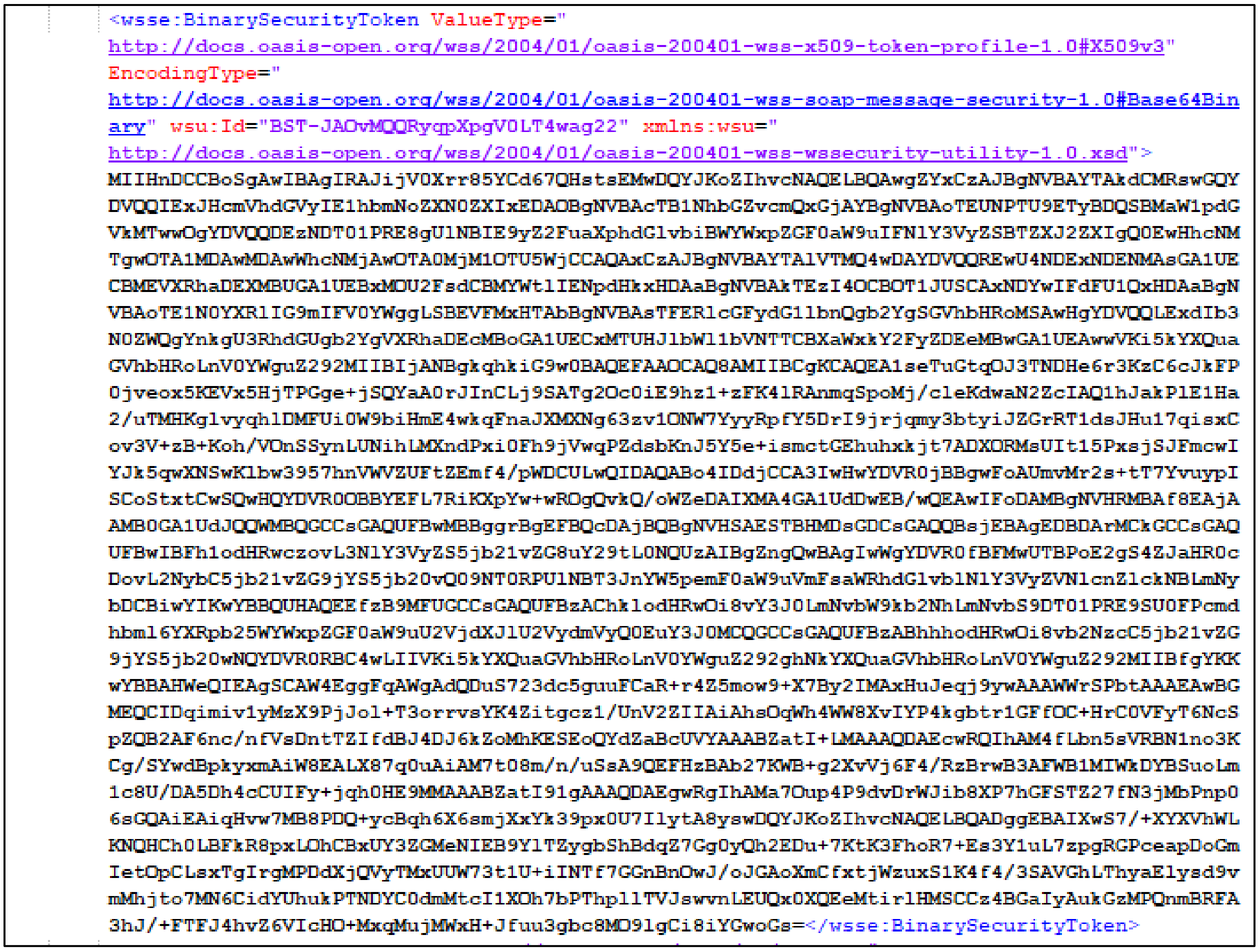
Figure 3: A Sample an EVV SOAP Envelope



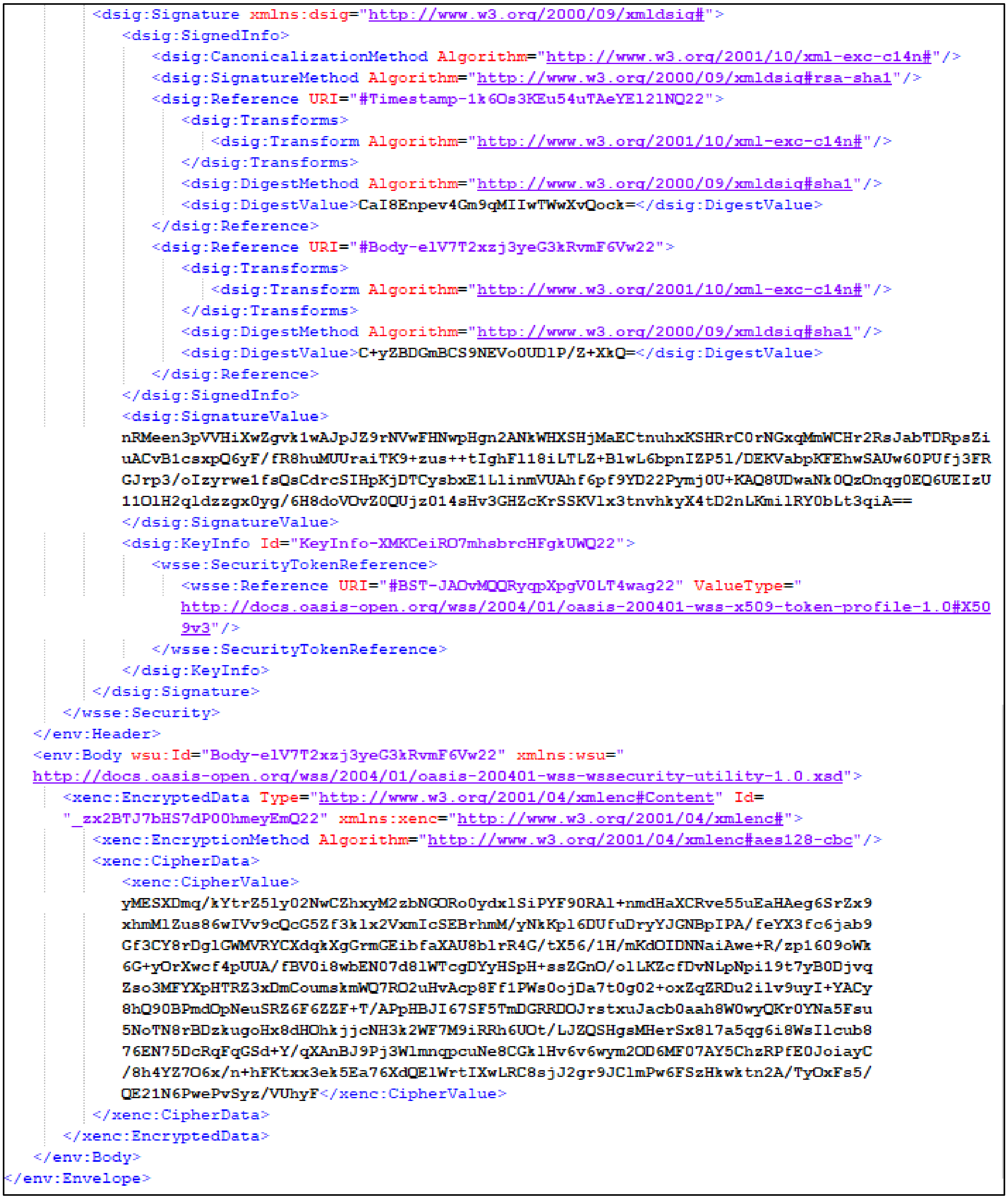
```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsa="
http://www.w3.org/2005/08/addressing">
   <env:Header>
      <wsa:MessageID>urn:4252e5ea-eae5-11e9-aa90-fd733cd5c9e7</wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
         <wsa:ReferenceParameters>
            <instra:tracking.compositeInstanceCreatedTime xmlns:instra="
            http://xmlns.oracle.com/sca/tracking/1.0">2019-10-09T16:36:37.674-06:00
            </instra:tracking.compositeInstanceCreatedTime>
         </wsa:ReferenceParameters>
      </wsa:ReplyTo>
      <wsa:FaultTo>
         <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
         <wsa:ReferenceParameters>
            <instra:tracking.compositeInstanceCreatedTime xmlns:instra="
            http://xmlns.oracle.com/sca/tracking/1.0">2019-10-09T16:36:37.674-06:00
            </instra:tracking.compositeInstanceCreatedTime>
         </wsa:ReferenceParameters>
      </wsa:FaultTo>
      <wsse:Security env:mustUnderstand="1" xmlns:wsse="
      http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <wsse:BinarySecurityToken ValueType="
         http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
         EncodingType="
         http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Bin
         ary" wsu:Id="BST-0pEYpsAeAwpvt3FMKwHAbw22" xmlns:wsu="
         http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
         MIIDbTCCAlWgAwIBAgIEWVHrvDANBgkqhkiG9w0BAQsFADBnMQswCQYDVQQGEwJVUzELMAkGA1UECBMCVVQxFzAVBgN
         VBAcTDlNhbHQgTGFrZSBDaXR5MQ8wDQYDVQQKEwZIZWFsdGgxDDAKBgNVBAsTA0RUUzETMBEGA1UEAxMKU09BUENsaW
         VudDAeFw0xOTEwMDcyMjI3MTFaFw0yOTA4MTUyMjI3MTFaMGcxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJVVDEXMBUGA
         1UEBxMOU2FsdCBMYWtlIENpdHkxDzANBgNVBAoTBkhlYWx0aDEMMAoGA1UECxMDRFRTMRMwEQYDVQQDEwpTT0FQQ2xp
         ZW50MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoU+ImidmHiUxvYkYdVuJYGF7YR0au3uPP1uhaw4RGI1
         jPA9LWaWHPlTsZf4beZ0mM3la223c8hfRMhYCpz44LxL9CnkpB6Dc6dIlSF+v/XdGH1ZEjIPVQLjNWcM/crVlWRE+nO
         exjaQP61rdyHZ49HTQ4+zPXuGFKoFdNUkutNbFZVwSySQkMvNuAeaB91WXgzOYmID8fPQZuH52Do9Ms5j4SQvONeYIr
         wzlhKJru4rwCF2fbD8s0l6lvj1dpenQIjhLvvOcwTPo3vvnFz5wxp724feaP0auzC42D0g8qfxUtfeXnkL71quTG/dc
         8pQTns5fmR6oqQAXqCJHu87kPQIDAQABoyEwHzAdBgNVHQ4EFgQU8wp2wi+q/8CZ4exC35dk1AAoWqEwDQYJKoZIhvc
         NAQELBQADggEBAAZU2+JziOgjZVxJ4t+7SN106Z20xyMz6Wb8NTCPq5kv38ING6dnTNedjbPLC89HBkt+OIA6eEMG7L
         QO98fuPiX194FImTm/1gyzkxhdS+22/LbgDKIOWGTK5NOyzjGq5racN6cpRNL+VzviGLuAgz82CrQHKUXjX90sxnfBz
         1OqySAH2Ux4tOnkoBVcZQe0yEQC2g3Fltu64lt2QkcsA4DYnR9VwMspkjLfCmsyhWeoYWGPtNliuJJbSglXO//72nPL
         /2Y4QnwUT7/aEu1lxH3pyerayHoM+hj+Dep5yO2+XXq6FODOB9jwvjEbnwkwZS81Pf9tJvZjgIfjWHdJZeM=
         </wsse:BinarySecurityToken>
```



```
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
      <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" xmlns:dsig="
      http://www.w3.org/2000/09/xmldsig#"/>
   </xenc:EncryptionMethod>
   <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <wsse:SecurityTokenReference>
         <wsse:Reference URI="#BST-OpEYpsAeAwpvt3FMKwHAbw22" ValueType="
         http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X50
         9v3"/>
      </wsse:SecurityTokenReference>
   </dsig:KeyInfo>
   <xenc:CipherData>
      <xenc:CipherValue>
      OI2YloRkz7/aXrdrBNizcPnhPro9hh4FF8ejMP3ig5PLEe9UF+Qj5aKKmgZwRBJb+1O7Y3b7N8hy
       Th2YYrDa2w5FUFcDRXiAnxDz5PWYDCVddr7upoL0Ldm6oRB0YJGKXjELvtFBzmpLJJx8XX7F/Y9G
       ABleTn7TD85mvYx8lsp3yaMer5Ka5H3YHtt8uWlXfFy9+7nz6Rte3sv+9IEwQWLYnI+mlA2sxLi4
       Uwb0iytJFCvwwSrAN8BZzoiWrvj7cDW4Kg2AbEoXpK8dQ/Ux+TkMF47WZbYdI6Rr5WimY5zo2kPU
       ILOlOuajUO7CClc41G+0DGuqSxPhxorJKR412Q==</xenc:CipherValue>
   </xenc:CipherData>
   <xenc:ReferenceList>
      <xenc:DataReference URI="#_zx2BTJ7bHS7dP00hmeyEmQ22"/>
   </xenc:ReferenceList>
</xenc:EncryptedKey>
<wsu:Timestamp wsu:Id="Timestamp-1k6Os3KEu54uTAeYE121NQ22" xmlns:wsu="
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <wsu:Created>2019-10-09T22:36:39Z</wsu:Created>
   <wsu:Expires>2019-10-13T09:56:39Z</wsu:Expires>
</wsu:Timestamp>
```



```
<wsse:BinarySecurityToken ValueType="
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
EncodingType="
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Bin
ary" wsu:Id="BST-JAOvMQQRyqpXpgV0LT4wag22" xmlns:wsu="
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
MIIHnDCCBoSgAwIBAgIRAJijV0Xrr85YCd67QHstsEMwDQYJKoZIhvcNAQELBQAwgZYxCzAJBgNVBAYTAkdCMRswGQY
DVQQIExJHcmVhdGVyIE1hbmNoZXN0ZXIxEDAOBgNVBAcTB1NhbGZvcmQxGjAYBgNVBAoTEUNPTU9ETyBDQSBMaW1pdG
VkMTwwOgYDVQQDEzNDT01PRE8gU1NBIE9yZ2FuaXphdGlvbiBWYWxpZGF0aW9uIFN1Y3VyZSBTZXJ2ZXIgQ0EwHhcNM
TgwOTA1MDAwMDAwWhcNMjAwOTA0MjM1OTU5WjCCAQAxCzAJBgNVBAYTAlVTMQ4wDAYDVQQREwU4NDExNDENMAsGA1UE
CBMEVXRhaDEXMBUGA1UEBxMOU2FsdCBMYWtlIENpdHkxHDAaBgNVBAkTEzI4OCBOT1JUSCAxNDYwIFdFU1QxHDAaBgN
VBAoTE1N0YXRlIG9mIFV0YWggLSBEVFMxHTAbBgNVBAsTFER1cGFydG1lbnQgb2YgSGVhbHRoMSAwHgYDVQQLExdIb3
N0ZWQgYnkgU3RhdGUgb2YgVXRhaDEcMBoGA1UECxMTUHJlbWl1bVNTTCBXaWxkY2FyZDEeMBwGA1UEAwwVKi5kYXQua
GVhbHRoLnV0YWguZ292MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1seTuGtqOJ3TNDHe6r3KzC6cJkFP
0jveox5KEVx5HjTPGge+jSQYaA0rJInCLj9SATg2Oc0iE9hz1+zFK41RAnmqSpoMj/cleKdwaN2ZcIAQ1hJakPlE1Ha
2/uTMHKglvyqhlDMFUi0W9biHmE4wkqFnaJXMXNg63zv1ONW7YyyRpfY5DrI9jrjqmy3btyiJZGrRT1dsJHu17qisxC
ov3V+zB+Koh/VOnSSynLUNihLMXndPxi0Fh9jVwqPZdsbKnJ5Y5e+ismctGEhuhxkjt7ADXORMsUIt15PxsjSJFmcwI
YJk5qwXNSwKlbw3957hnVWVZUFtZEmf4/pWDCULwQIDAQABo4IDdjCCA3IwHwYDVR0jBBgwFoAUmvMr2s+tT7YvuypI
SCoStxtCwSQwHQYDVR0OBBYEFL7RiKXpYw+wROgQvkQ/oWZeDAIXMA4GA1UdDwEB/wQEAwIFoDAMBgNVHRMBAf8EAjA
AMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjBQBgNVHSAESTBHMDsGDCsGAQQBsjEBAgEDBDArMCkGCCsGAQ
UFBwIBFh1odHRwczovL3NlY3VyZS5jb21vZG8uY29tL0NQUzAIBgZngQwBAgIwWgYDVR0fBFMwUTBPoE2gS4ZJaHR0c
DovL2NybC5jb21vZG9jYS5jb20vQ09NT0RPU1NBT3JnYW5pemF0aW9uVmFsaWRhdGlvblN1Y3VyZVN1cnZlckNBLmNy
bDCBiwYIKwYBBQUHAQEEfzB9MFUGCCsGAQUFBzAChklodHRwOi8vY3J0LmNvbW9kb2NhLmNvbS9DT01PRE9SU0FPcmd
hbml6YXRpb25WYWxpZGF0aW9uU2VjdXJlU2VydmVyQ0EuY3J0MCQGCCsGAQUFBzABhhhodHRwOi8vb2NzcC5jb21vZG
9jYS5jb20wNQYDVR0RBC4wLIIVKi5kYXQuaGVhbHRoLnV0YWguZ292ghNkYXQuaGVhbHRoLnV0YWguZ292MIIBfgYKK
wYBBAHWeQIEAgSCAW4EggFqAWgAdQDuS723dc5guuFCaR+r4Z5mow9+X7By2IMAxHuJeqj9ywAAAWWrSPbtAAAEAwBG
MEQCIDqimiv1yMzX9PjJol+T3orrvsYK4Zitgcz1/UnV2ZIIAiAhsOqWh4WW8XvIYP4kgbtr1GFfOC+HrC0VFyT6NcS
pZQB2AF6nc/nfVsDntTZIfdBJ4DJ6kZoMhKESEoQYdZaBcUVYAAABZatI+LMAAAQDAEcwRQIhAM4fLbn5sVRBN1no3K
Cg/SYwdBpkyxmAiW8EALX87q0uAiAM7t08m/n/uSsA9QEFHzBAb27KWB+g2XvVj6F4/RzBrwB3AFWB1MIWkDYBSuoLm
1c8U/DA5Dh4cCUIFy+jqh0HE9MMAAABZatI91gAAAQDAEgwRgIhAMa7Oup4P9dvDrWJib8XP7hGFSTZ27fN3jMbPnp0
6sGQAiEAiqHvw7MB8PDQ+ycBqh6X6smjXxYk39px0U7IlytA8yswDQYJKoZIhvcNAQELBQADggEBAIXwS7/+XYXVhWL
KNQHCh0LBFkR8pxLOhCBxUY3ZGMeNIEB9YlTZygbShBdqZ7Gg0yQh2EDu+7KtK3FhoR7+Es3Y1uL7zpgRGPceapDoGm
IetOpCLsxTgIrgMPDdXjQVyTMxUUW73t1U+iINTf7GGnBnOwJ/oJGAoXmCfxtjWzuxS1K4f4/3SAVGhLThyaElysd9v
mMhjto7MN6CidYUhukPTNDYC0dmMtcI1XOh7bPThpllTVJswvnLEUQx0XQEeMtirlHMSCCz4BGaIyAukGzMPQnmBRFA
3hJ/+FTFJ4hvZ6VIcHO+MxqMujMWxH+Jfuu3gbc8MO91gCi8iYGwoGs=</wsse:BinarySecurityToken>
```



```
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
        <dsig:SignedInfo>
          <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <dsig:Reference URI="#Timestamp-1k6Os3KEu54uTAeYE121NQ22">
            <dsig:Transforms>
              <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </dsig:Transforms>
            <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue>CaI8Enpev4Gm9qMIIwTWwXvQock=</dsig:DigestValue>
          </dsig:Reference>
          <dsig:Reference URI="#Body-elV7T2xzj3yeG3kRvmF6Vw22">
            <dsig:Transforms>
              <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </dsig:Transforms>
            <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue>C+yZBDGmBCS9NEVo0UD1P/Z+XkQ=</dsig:DigestValue>
          </dsig:Reference>
        </dsig:SignedInfo>
        <dsig:SignatureValue>
        nRMeen3pVVHiXwZgvk1wAJpJZ9rNVwFHNwpHgn2ANkWHXSHjMaECtnuhxKSHRrC0rNGxqMmWCHr2RsJabTDRpsZi
        uACvB1csxpQ6yF/fR8huMUUraiTK9+zus++tIghFl18iLTLZ+BlwL6bpnIZP5l/DEKVabpKFEhwSAUw60PUfj3FR
        GJrp3/oIzyrwe1fsQsCdrcSIHpKjDTCysbxE1LlinmVUAhf6pf9YD22Pymj0U+KAQ8UDwaNk0QzOnqg0EQ6UEIzU
        11OlH2qldzzgx0yg/6H8doVOvZ0QUjz014sHv3GHZcKrSSKVlx3tnvhkyX4tD2nLKmilRY0bLt3qiA==
        </dsig:SignatureValue>
        <dsig:KeyInfo Id="KeyInfo-XMKCeiRO7mhsbrcHFgkUWQ22">
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#BST-JAOvMQQRyqpXpgV0LT4wag22" ValueType="
            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X50
            9v3"/>
          </wsse:SecurityTokenReference>
        </dsig:KeyInfo>
      </dsig:Signature>
    </wsse:Security>
  </env:Header>
  <env:Body wsu:Id="Body-elV7T2xzj3yeG3kRvmF6Vw22" xmlns:wsu="
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
    <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Content" Id=
    "_zx2BTJ7bHS7dP00hmeyEmQ22" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <xenc:CipherData>
        <xenc:CipherValue>
        yMESXDmq/kYtrZ5ly02NwCZhxyM2zbNGORo0ydxlSiPYF90RAl+nmdHaXCRve55uEaHAeg6SrZx9
        xhmMlZus86wIVv9cQcG5Zf3klx2VxmIcSEBrhmM/yNkKpl6DUfuDryYJGNBpIPA/feYX3fc6jab9
        Gf3CY8rDglGWMVRYCXdqkXgGrmGEibfaXAU8blrR4G/tX56/1H/mKdOIDNNaiAwe+R/zp1609oWk
        6G+yOrXwcf4pUUA/fBV0i8wbEN07d8lWTcgDYyHSpH+ssZGnO/olLKZcfDvNLpNpi19t7yB0Djvq
        Zso3MFYXpHTRZ3xDmCoumskmWQ7RO2uHvAcp8Ff1PWs0ojDa7t0g02+oxZqZRDu2ilv9uyI+YACy
        8hQ90BPmdOpNeuSRZ6F6ZZF+T/APpHBJI67SF5TmDGRRDOJrstxuJacb0aah8W0wyQKr0YNa5Fsu
        5NoTN8rBDzkugoHx8dHOhkjjcNH3k2WF7M9iRRh6UOt/LJZQSHgsMHerSx8l7a5qg6i8WsIlcub8
        76EN75DcRqFqGSd+Y/qXAnBJ9Pj3WlmnqpcuNe8CGklHv6v6wym2OD6MF07AY5ChzRPfE0JoiayC
        /8h4YZ7O6x/n+hFKtxx3ek5Ea76XdQElWrtIXwLRC8sjJ2gr9JClmPw6FSzHkwktn2A/TyOxFs5/
        QE21N6PwePvSyz/VUhyF</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </env:Body>
</env:Envelope>
```

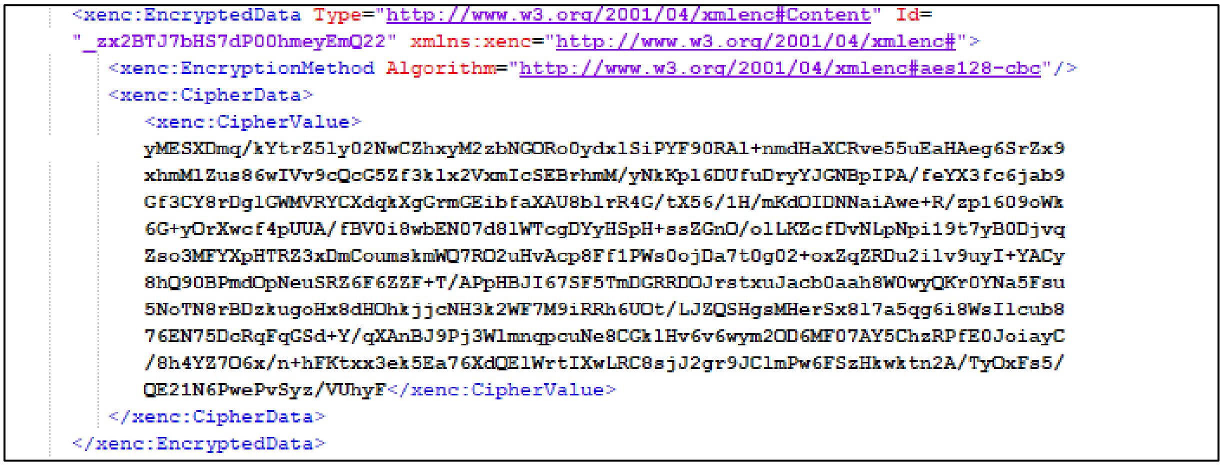
4.2.1.4. Digital Certificates

Digital certificates bind digital information to physical identities and provide non-repudiation and data integrity. Before you begin the enrollment process, each entity should obtain one valid digital certificate issued by an approved certificate authority (CA) and sends to Utah Medicaid to be stored in the database; the State only recognizes and accepts submissions from providers who have a valid certificate in the system. There should only be one certificate per submitting provider, and it should not be used by any other service.

A provider's authorized representative obtains a digital certificate from their Information Technology (I.T.) department and securely sends the certificate to Utah Medicaid's EVV contact person who is responsible for documenting and forwarding the certificate to the responsible team at Department of Technology Services (DTS) for installation. The WSDL will then be created and communicated by the State to the provider's authorized official to be set up for file exchanges.

4.3. SOAP Body

The SOAP Body contains encrypted payload of the submission Data File by the transmitter application. The <EncryptionMethod> element identifies the algorithm used to encrypt the data file as shown below.



```
<xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Content" Id=
"_zx2BTJ7bHS7dP00hmeyEmQ22" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
   <xenc:CipherData>
      <xenc:CipherValue>
      yMESXDmq/kYtrZ5ly02NwCZhxyM2zbNGORo0ydxlSiPYF90RAl+nmdHaXCRve55uEaHAeg6SrZx9
      xhmMlZus86wIVv9cQcG5Zf3klx2VxmIcSEBrhmM/yNkKpl6DUfuDryYJGNBpIPA/feYX3fc6jab9
      Gf3CY8rDglGWMVRYCXdqkXgGrmGEibfaXAU8blrR4G/tX56/1H/mKdOIDNNaiAwe+R/zp1609oWk
      6G+yOrXwcf4pUUA/fBV0i8wbEN07d8lWTcgDYyHSpH+ssZGnO/olLKZcfDvNLpNpi19t7yB0Djvq
      Zso3MFYXpHTRZ3xDmCoumskmWQ7RO2uHvAcp8Ff1PWs0ojDa7t0g02+oxZqZRDu2ilv9uyI+YACy
      8hQ90BPmdOpNeuSRZ6F6ZZF+T/APpHBJI67SF5TmDGRRDOJrstxuJacb0aah8W0wyQKr0YNa5Fsu
      5NoTN8rBDzkugoHx8dHOhkjjcNH3k2WF7M9iRRh6UOt/LJZQSHgsMHerSx817a5qg6i8WsIlcub8
      76EN75DcRqFqGSd+Y/qXAnBJ9Pj3WlmnqpcuNe8CGklHv6v6wym2OD6MF07AY5ChzRPfE0JoiayC
      /8h4YZ7O6x/n+hFKtxx3ek5Ea76XdQElWrtIXwLRC8sjJ2gr9JClmPw6FSzHkwktn2A/TyOxFs5/
      QE21N6PwePvSyz/VUhyF</xenc:CipherValue>
   </xenc:CipherData>
</xenc:EncryptedData>
```

4.4. Guidelines for Composing EVV Data File

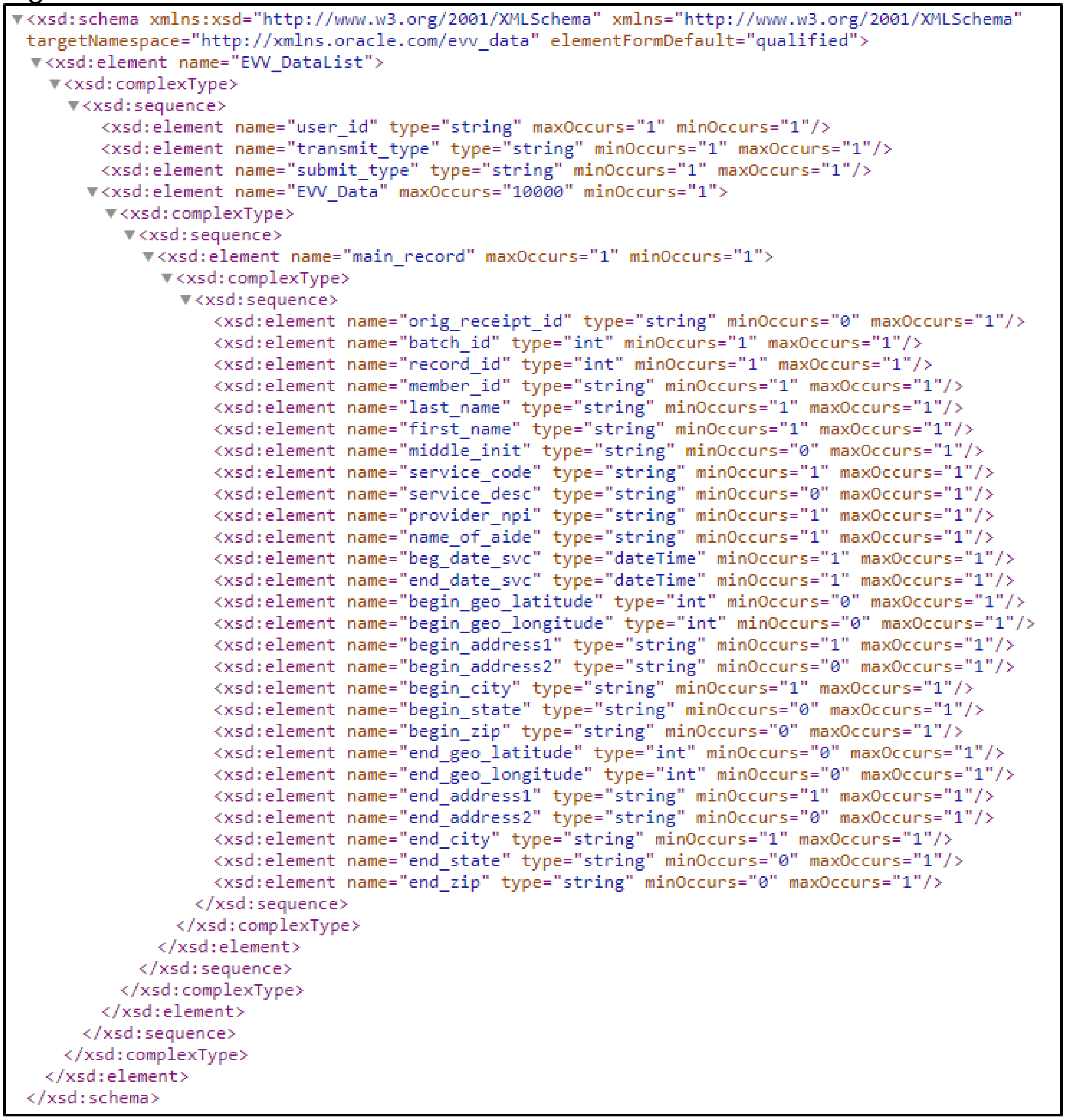
Below are general guidelines for composing the submission Data File:

1. The data file can only contain valid uncompressed and unencrypted XML.
2. The data file must contain at least one (1) and cannot exceed 10,000 records per transmission.

4.5. Structure of EVV Submission Data File

The submission data file uses the XML schema as display below.

Figure 4: EVV Data File XML Schema



```
▼<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://xmlns.oracle.com/evv_data" elementFormDefault="qualified">
  ▼<xsd:element name="EVV_DataList">
    ▼<xsd:complexType>
      ▼<xsd:sequence>
          <xsd:element name="user_id" type="string" maxOccurs="1" minOccurs="1"/>
          <xsd:element name="transmit_type" type="string" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="submit_type" type="string" minOccurs="1" maxOccurs="1"/>
        ▼<xsd:element name="EVV_Data" maxOccurs="10000" minOccurs="1">
          ▼<xsd:complexType>
            ▼<xsd:sequence>
              ▼<xsd:element name="main_record" maxOccurs="1" minOccurs="1">
                ▼<xsd:complexType>
                  ▼<xsd:sequence>
                      <xsd:element name="orig_receipt_id" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="batch_id" type="int" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="record_id" type="int" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="member_id" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="last_name" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="first_name" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="middle_init" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="service_code" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="service_desc" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="provider_npi" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="name_of_aide" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="beg_date_svc" type="dateTime" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="end_date_svc" type="dateTime" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="begin_geo_latitude" type="int" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="begin_geo_longitude" type="int" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="begin_address1" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="begin_address2" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="begin_city" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="begin_state" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="begin_zip" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="end_geo_latitude" type="int" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="end_geo_longitude" type="int" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="end_address1" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="end_address2" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="end_city" type="string" minOccurs="1" maxOccurs="1"/>
                      <xsd:element name="end_state" type="string" minOccurs="0" maxOccurs="1"/>
                      <xsd:element name="end_zip" type="string" minOccurs="0" maxOccurs="1"/>
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

4.6. Data File XML Elements

The following table provides details of XML elements that the XML schema requires in the EVV data file of the SOAP message when transmitting information to Utah Medicaid.

Table 7: EVV Data File XML Elements

Element Name	Description	Type	Required	Min. Occurs	Max. Occurs
EVV_DataList	Start of the EVV data file		Yes	1	1
user_id	Provider or person who creates the data file	VARCHAR (1-60)	Yes	1	1
transmit_type	Transmission Type	VARCHAR (1) *Value:* **B** (Batch)	Yes	1	1
submit_type	Submission Type	VARCHAR (1) *Value:* **N** (New), **C** (Correction)	Yes	1	1
EVV_Data	Start of an EVV record		Yes	1	10,000
main_record	Start of EVV record		Yes	1	1
orig_receipt_id	Receipt_id of the record to be replaced. Required if submit a Correction record.	VARCHAR (32)	No	0	1
batch_id	Batch_id # in the data file	VARCHAR (1-10)	Yes	1	1
record_id	Record_id # in the batch	VARCHAR (1-10)	Yes	1	1
member_id	Member Medicaid ID	VARCHAR (10)	Yes	1	1
last_name	Member last name	VARCHAR (1-255)	Yes	1	1

Element Name	Description	Type	Required	Min. Occurs	Max. Occurs
first_name	Member first name	VARCHAR (1-255)	Yes	1	1
middle_init	Member Middle Init	VARCHAR (1-255)	No	1	1
service_code	HCPCS/CPT code or DSPD service code	VARCHAR (1-5)	Yes	1	1
service_desc	Description of Service	VARCHAR (1-255)	No	1	1
provider_npi	Provider NPI or PRISM ID	VARCHAR (1-12)	Yes	1	1
name_of_aid	Name of person providing service	VARCHAR (1-255)	Yes	1	1
beg_date_svc	Begin date of service	DATE/TIME *Format:* YYYY-MM-DDTHH:MM:SS Example: (2019-09-01T10:15:00)	Yes	1	1
end_date_svc	End date of service	DATE/TIME *Format:* YYYY-MM-DDTHH:MM:SS Example: (2019-09-01T15:30:00)	Yes	1	1
begin_geo_latitude	Latitude coordinate	NUMBER (38, 10) Example: 28.523	No*	0	1

Element Name	Description	Type	Required	Min. Occurs	Max. Occurs
begin_geo_longitude	Longitude coordinate	NUMBER (38, 10) Example: 80.683	No*	0	1
begin_address1	Address where service was provided	VARCHAR (1-255)	Yes*	1	1
begin_address2	PO Box, apartment number, etc.	VARCHAR (1-255)	No	0	1
begin_city	City where service was provided	VARCHAR (1-60)	Yes*	1	1
begin_state	State where service was provided	VARCHAR (1-20)	No	0	1
begin_zip	Code where service was provided	VARCHAR (1-10)	No	0	1
end_geo_latitude	Latitude coordinate	NUMBER (38, 10) Example: 28.523	No	0	1
end_geo_longitude	Longitude coordinate	NUMBER (38, 10) Example: 80.683	No	0	1
end_address1	Address where service was provided	VARCHAR (1-255)	Yes	1	1
end_address2	PO Box, apartment number, etc.	VARCHAR (1-255)	No	0	1

Element Name	Description	Type	Required	Min. Occurs	Max. Occurs
end_city	City where service was provided	VARCHAR (1-60)	Yes	1	1
end_state	State where service was provided	VARCHAR (1-20)	No	0	1
end_zip	Code where service was provided	VARCHAR (1-10)	No	0	1

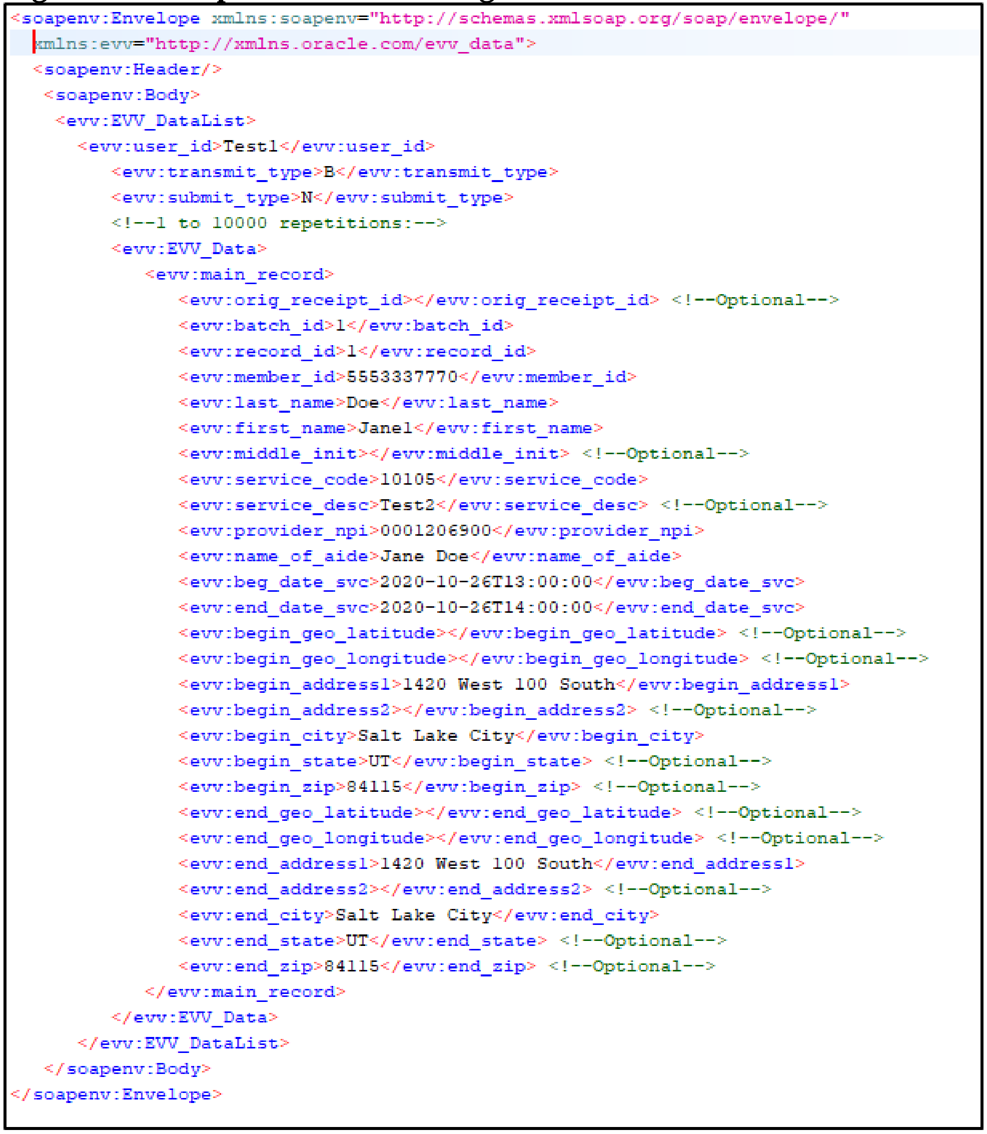
NOTE: ***Either begin_address1/begin_city OR begin_geo_latitude/begin_geo_longitude must be present**

4.7. Examples of Data File SOAP messages:

Providers can send new or correction to replace previously accepted submission. If a replacement is submitted, the receipt ID, batch ID and record ID of such records to be replaced must be provided in the correction file.

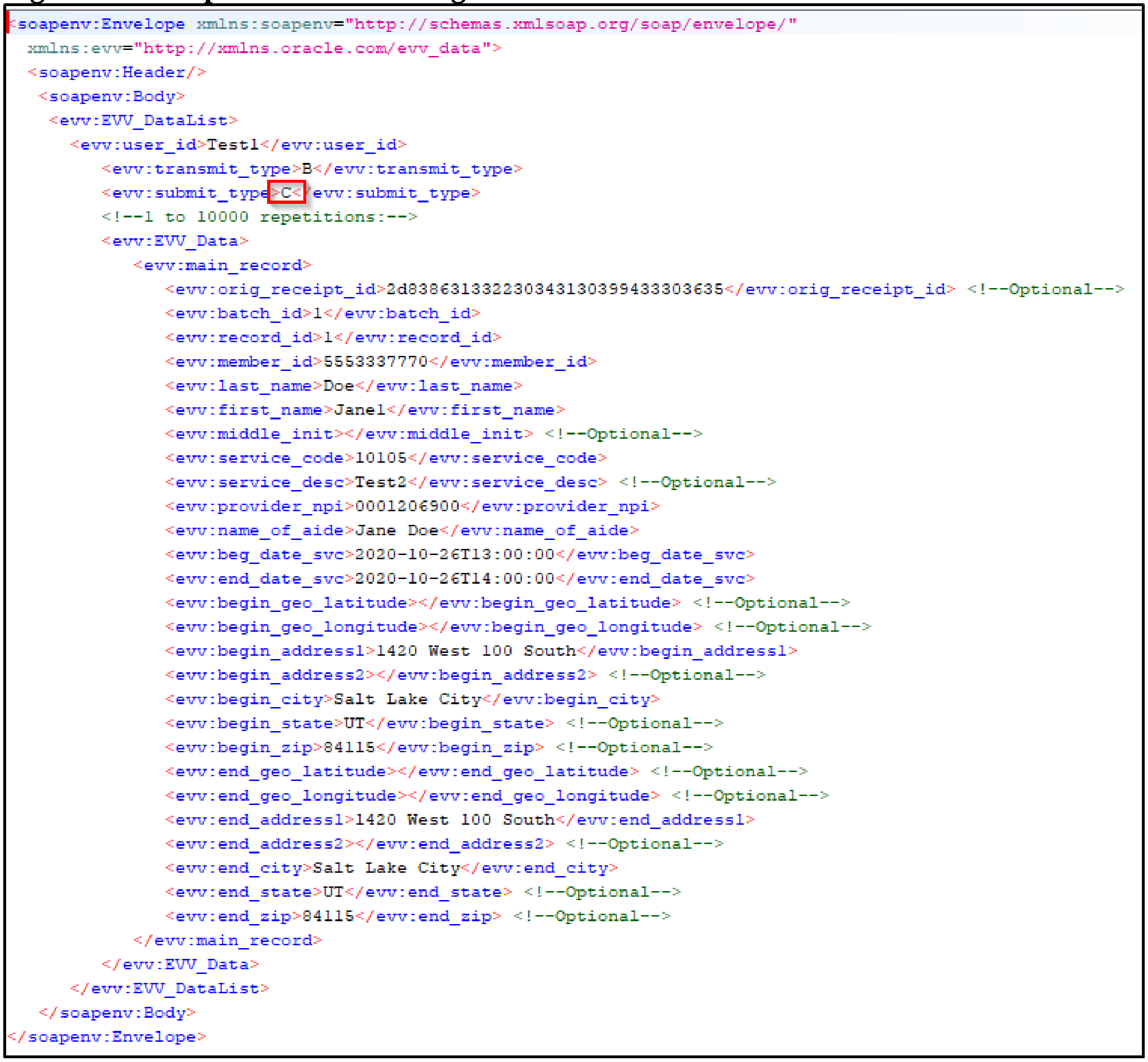
Below are some examples of SOAP messages for transmitting data file for a new and a correction EVV record in a submission batch.

Figure 5: Example of SOAP Message of a New EVV Record Batch



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:evv="http://xmlns.oracle.com/evv_data">
  <soapenv:Header/>
   <soapenv:Body>
    <evv:EVV_DataList>
      <evv:user_id>Test1</evv:user_id>
         <evv:transmit_type>B</evv:transmit_type>
         <evv:submit_type>N</evv:submit_type>
         <!--1 to 10000 repetitions:-->
         <evv:EVV_Data>
            <evv:main_record>
               <evv:orig_receipt_id></evv:orig_receipt_id> <!--Optional-->
               <evv:batch_id>1</evv:batch_id>
               <evv:record_id>1</evv:record_id>
               <evv:member_id>5553337770</evv:member_id>
               <evv:last_name>Doe</evv:last_name>
               <evv:first_name>Jane1</evv:first_name>
               <evv:middle_init></evv:middle_init> <!--Optional-->
               <evv:service_code>10105</evv:service_code>
               <evv:service_desc>Test2</evv:service_desc> <!--Optional-->
               <evv:provider_npi>0001206900</evv:provider_npi>
               <evv:name_of_aide>Jane Doe</evv:name_of_aide>
               <evv:beg_date_svc>2020-10-26T13:00:00</evv:beg_date_svc>
               <evv:end_date_svc>2020-10-26T14:00:00</evv:end_date_svc>
               <evv:begin_geo_latitude></evv:begin_geo_latitude> <!--Optional-->
               <evv:begin_geo_longitude></evv:begin_geo_longitude> <!--Optional-->
               <evv:begin_address1>1420 West 100 South</evv:begin_address1>
               <evv:begin_address2></evv:begin_address2> <!--Optional-->
               <evv:begin_city>Salt Lake City</evv:begin_city>
               <evv:begin_state>UT</evv:begin_state> <!--Optional-->
               <evv:begin_zip>84115</evv:begin_zip> <!--Optional-->
               <evv:end_geo_latitude></evv:end_geo_latitude> <!--Optional-->
               <evv:end_geo_longitude></evv:end_geo_longitude> <!--Optional-->
               <evv:end_address1>1420 West 100 South</evv:end_address1>
               <evv:end_address2></evv:end_address2> <!--Optional-->
               <evv:end_city>Salt Lake City</evv:end_city>
               <evv:end_state>UT</evv:end_state> <!--Optional-->
               <evv:end_zip>84115</evv:end_zip> <!--Optional-->
            </evv:main_record>
         </evv:EVV_Data>
      </evv:EVV_DataList>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 6: Example of SOAP Message of A Correction EVV Batch



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:evv="http://xmlns.oracle.com/evv_data">
  <soapenv:Header/>
   <soapenv:Body>
    <evv:EVV_DataList>
      <evv:user_id>Test1</evv:user_id>
         <evv:transmit_type>B</evv:transmit_type>
         <evv:submit_type>C</evv:submit_type>
         <!--1 to 10000 repetitions:-->
         <evv:EVV_Data>
            <evv:main_record>
               <evv:orig_receipt_id>2d838631332230343130399433303635</evv:orig_receipt_id> <!--Optional-->
               <evv:batch_id>1</evv:batch_id>
               <evv:record_id>1</evv:record_id>
               <evv:member_id>5553337770</evv:member_id>
               <evv:last_name>Doe</evv:last_name>
               <evv:first_name>Jane1</evv:first_name>
               <evv:middle_init></evv:middle_init> <!--Optional-->
               <evv:service_code>10105</evv:service_code>
               <evv:service_desc>Test2</evv:service_desc> <!--Optional-->
               <evv:provider_npi>0001206900</evv:provider_npi>
               <evv:name_of_aide>Jane Doe</evv:name_of_aide>
               <evv:beg_date_svc>2020-10-26T13:00:00</evv:beg_date_svc>
               <evv:end_date_svc>2020-10-26T14:00:00</evv:end_date_svc>
               <evv:begin_geo_latitude></evv:begin_geo_latitude> <!--Optional-->
               <evv:begin_geo_longitude></evv:begin_geo_longitude> <!--Optional-->
               <evv:begin_address1>1420 West 100 South</evv:begin_address1>
               <evv:begin_address2></evv:begin_address2> <!--Optional-->
               <evv:begin_city>Salt Lake City</evv:begin_city>
               <evv:begin_state>UT</evv:begin_state> <!--Optional-->
               <evv:begin_zip>84115</evv:begin_zip> <!--Optional-->
               <evv:end_geo_latitude></evv:end_geo_latitude> <!--Optional-->
               <evv:end_geo_longitude></evv:end_geo_longitude> <!--Optional-->
               <evv:end_address1>1420 West 100 South</evv:end_address1>
               <evv:end_address2></evv:end_address2> <!--Optional-->
               <evv:end_city>Salt Lake City</evv:end_city>
               <evv:end_state>UT</evv:end_state> <!--Optional-->
               <evv:end_zip>84115</evv:end_zip> <!--Optional-->
            </evv:main_record>
         </evv:EVV_Data>
      </evv:EVV_DataList>
   </soapenv:Body>
</soapenv:Envelope>
```

4.8. API Data Validation and Error Message

The table below provides information regarding the front-end data input validation and error messages when data input failed the front-end validation.

Table 8: API Input Data Validations and Error Messages

Input Field	Data Input Validations	Error Message
user_id	● Not null	● Missing User ID
transmit_type	● Not null ● Value equals “B”	● Missing or invalid transmit_type

Input Field	Data Input Validations	Error Message
		● Missing or invalid transmit_type
submit_type	● Not null ● Value equals "N" or "C"	● Missing or invalid submit type ● Missing or invalid submit type
orig_receipt_id	● If transmit_type is "C", and this field cannot be null	● Missing required field
batch_id	● Not Null ● Exceed limit data length	● Missing required field ● Data Fields Exceeded Limit
record_id	● Not null ● Exceed limit data length	● Missing required field ● Data Fields Exceeded Limit
member_id	● Not Null ● Exceed limit data length	● Missing required field ● Data Fields Exceeded Limit
last_name	● Not Null	● Missing required field
first_name	● Not Null	● Missing required field
service_code	● Not Null ● Exceed limit data length	● Missing required field ● Data Fields Exceeded Limit
provider_npi	● Not Null ● Numeric value ● Exceed limit data length	● Missing required field ● Invalid Data Types ● Data Fields Exceeded Limit
name_of_aide	● Not Null	● Missing required field
beg_date_svc	● YYYY-MM-DDTHH:MM:SS Example 2019-09-29T14:30:00 ● Begin date/time not greater than End Date/End Time ● Not exceed 365 days ● Cannot be a future date ● Individual service duration cannot exceed 24 hours	● Invalid Date/Time format. Please re-submit using YYYY-MM-DDTHH:MM:SS format ● End Service date and time should be greater than Begin Service data and time. ● Service date/time cannot exceed 1 year from Date of Service. ● Service date/time cannot be a future date/time. ● Service duration cannot exceed 24 hours.

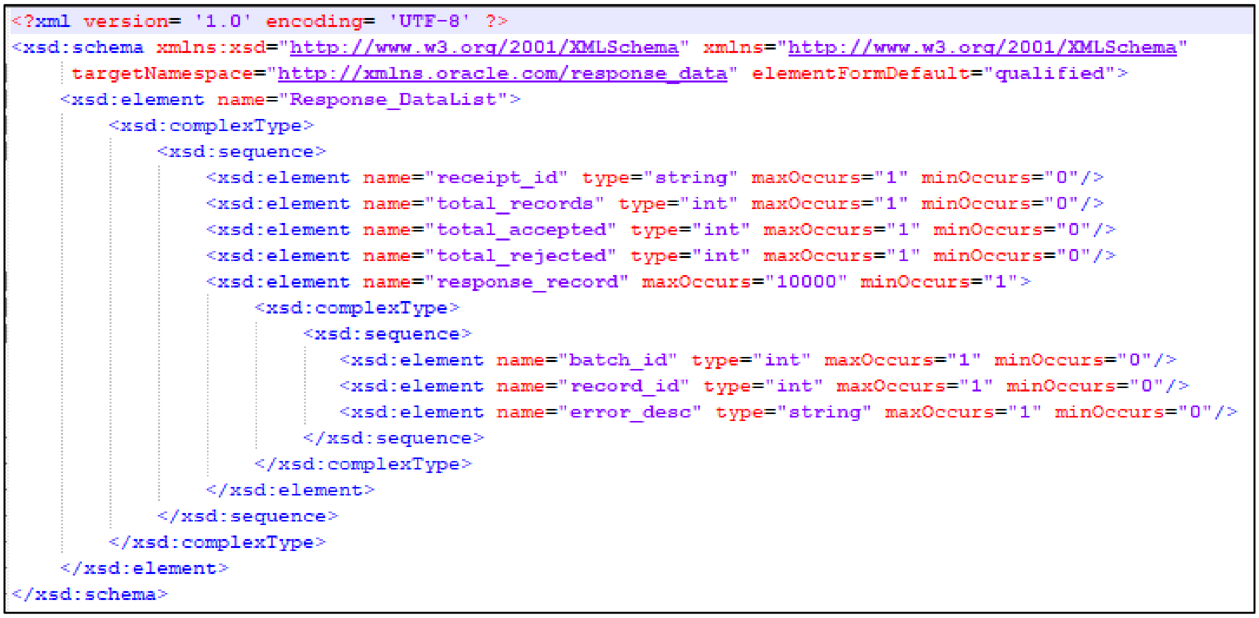
Input Field	Data Input Validations	Error Message
end_date_svc	● YYYY-MM-DDTHH:MM:SS Example 2019-09-29T14:30:00 ● Begin date/time not greater than End Date/End Time ● Not exceed 365 days ● Cannot be a furfure date ● Individual service duration cannot exceed 24 hours	● Invalid Date/Time format. Please re-submit using YYYY-MM-DDTHH:MM:SS format ● End Service date and time should be greater than Begin Service data and time. ● Service date/time cannot exceed 1 year from Date of Service. ● Service date/time cannot be a future date/time. ● Service duration cannot exceed 24 hours.
begin_address1	● Not Null (without begin_geo_latitude and begin_geo_longitude) ● Null permitted (with begin_geo_latitude and begin_geo_longitude	● Missing required field
begin_city	● Not Null (without begin_geo_latitude and begin_geo_longitude) ● Null permitted (with begin_geo_latitude and begin_geo_longitude	● Missing required field

5. API Submission Acknowledgement XML Schema

Once the provider’s transmission is authenticated and authorized, the data in the transmission is processed in sequence, record by record. The submission is accepted if all records in the batch meet data input validations, otherwise, the entire file is returned to the provider for correction and re-submits.

If some of the records in the batch are accepted and some failed, the acknowledgment SOAP message will indicate the number of record submitted in the batch, the total numbers of records were accepted and the total numbers of records are rejected, along with the rejected record detail (including batch-id and record-id where the error message is located in the provider’s file)

Figure 7: Submission Acknowledgement XML Schema



```
<?xml version= '1.0' encoding= 'UTF-8' ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://xmlns.oracle.com/response_data" elementFormDefault="qualified">
    <xsd:element name="Response_DataList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="receipt_id" type="string" maxOccurs="1" minOccurs="0"/>
                <xsd:element name="total_records" type="int" maxOccurs="1" minOccurs="0"/>
                <xsd:element name="total_accepted" type="int" maxOccurs="1" minOccurs="0"/>
                <xsd:element name="total_rejected" type="int" maxOccurs="1" minOccurs="0"/>
                <xsd:element name="response_record" maxOccurs="10000" minOccurs="1">
                    <xsd:complexType>
                        <xsd:sequence>
                           <xsd:element name="batch_id" type="int" maxOccurs="1" minOccurs="0"/>
                           <xsd:element name="record_id" type="int" maxOccurs="1" minOccurs="0"/>
                           <xsd:element name="error_desc" type="string" maxOccurs="1" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Table 9: EVV Acknowledgement XML Elements

Element Name	Description	Type	Max. Occurs	Min. Occurs
Response_DataList	Start of the response message		1	0
receipt_id	Utah Medicaid’s Receipt ID for the provider’s submission	Num (32)	1	0
total_records	Counts of number records included in the submitting batch	Num (1-10)	1	0
total_Accepted	Counts of total accepted EVV records in the submitting batch	Num (1-10)	1	0
total_rejected	Counts of total rejected EVV records in the submitting batch	Num (1-10)	1	0

Element Name	Description	Type	Max. Occurs	Min. Occurs
response_record	Start of identified error EVV record in the batch			
batch_id	The provider provided batch ID in the submitting file, which the error record is located	Num (1-10)	1	0
record_id	The provider provided record ID in the submitting file, which the error record is located	Num (1-10)	1	0
error_desc	Error message for the rejected record	VARCHAR (1-255)	1	1

6. Transmitting API Correction/ Replacement EVV Records

Providers can transmit a correction batch of EVV data; all need to be in its own batch with transmission type indicates as “C” and the original receipt ID, batch ID and record ID must be provided for each record within the batch, otherwise, the record will be rejected.

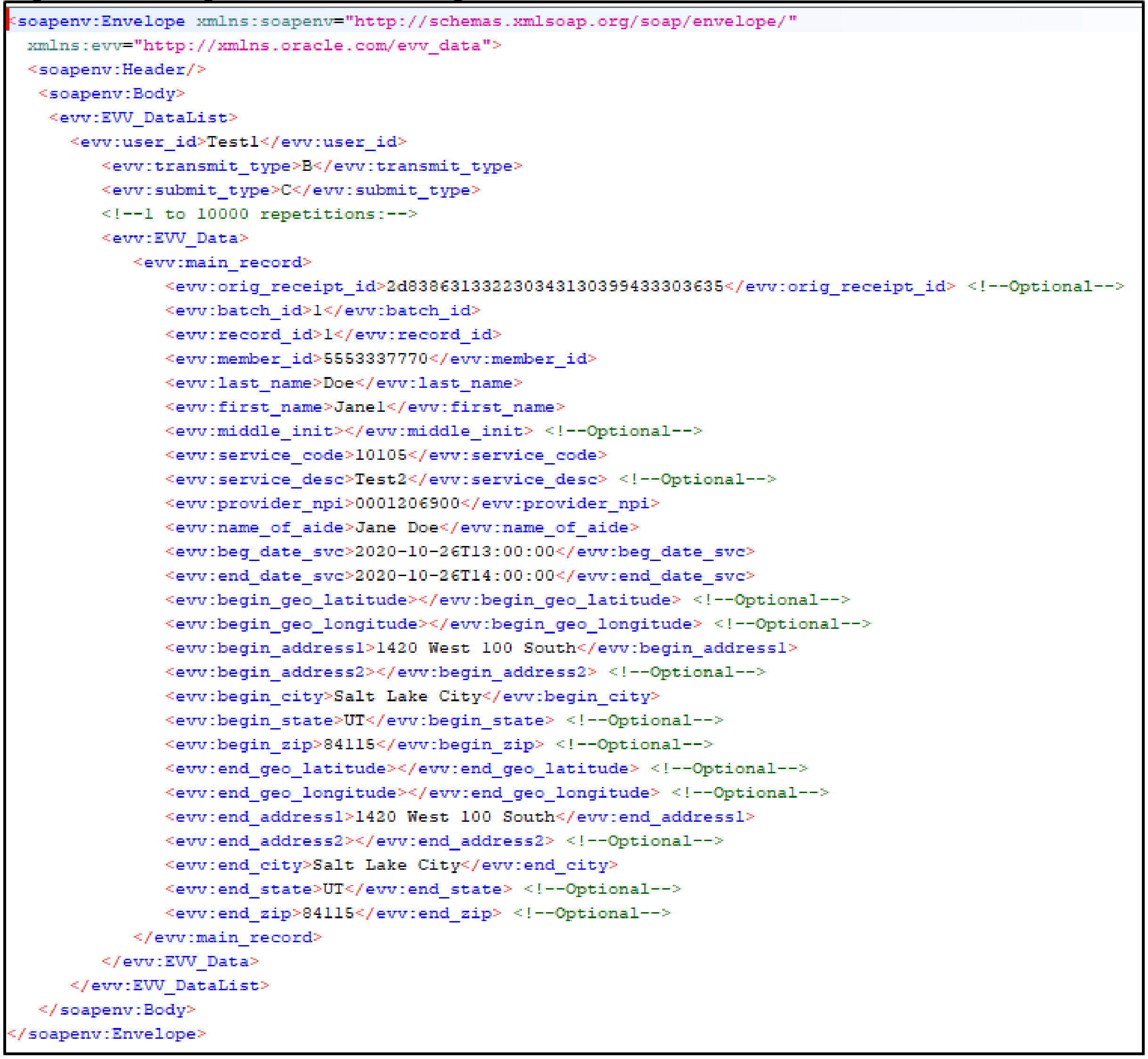
6.1. Correct and Replace original record that were rejected

If an EVV record is rejected during the first submission, such record it is returned to the provider to make the necessary correction and resubmit as a new record.

6.2. Correct and Replace original record that was accepted

If an EVV record was accepted during the record’s first submission and the provider needs to make a correction to replace this original at a later date, the original’s Receipt-ID, Batch-ID and Record-ID are required in the correction file.

Figure 8: Example of SOAP Message of A Correction EVV Batch

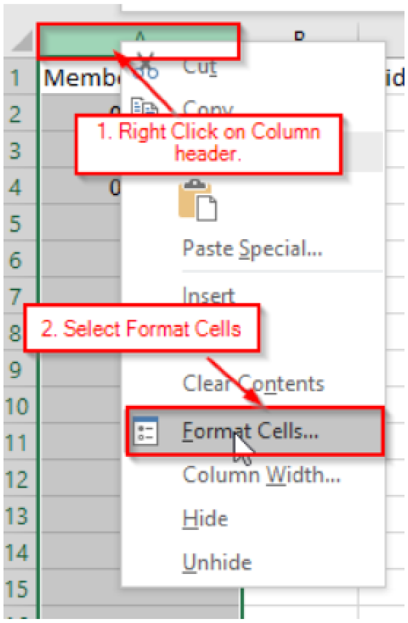


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:evv="http://xmlns.oracle.com/evv_data">
  <soapenv:Header/>
   <soapenv:Body>
    <evv:EVV_DataList>
      <evv:user_id>Test1</evv:user_id>
         <evv:transmit_type>B</evv:transmit_type>
         <evv:submit_type>C</evv:submit_type>
         <!--1 to 10000 repetitions:-->
         <evv:EVV_Data>
            <evv:main_record>
               <evv:orig_receipt_id>2d838631332230343130399433303635</evv:orig_receipt_id> <!--Optional-->
               <evv:batch_id>1</evv:batch_id>
               <evv:record_id>1</evv:record_id>
               <evv:member_id>5553337770</evv:member_id>
               <evv:last_name>Doe</evv:last_name>
               <evv:first_name>Jane1</evv:first_name>
               <evv:middle_init></evv:middle_init> <!--Optional-->
               <evv:service_code>10105</evv:service_code>
               <evv:service_desc>Test2</evv:service_desc> <!--Optional-->
               <evv:provider_npi>0001206900</evv:provider_npi>
               <evv:name_of_aide>Jane Doe</evv:name_of_aide>
               <evv:beg_date_svc>2020-10-26T13:00:00</evv:beg_date_svc>
               <evv:end_date_svc>2020-10-26T14:00:00</evv:end_date_svc>
               <evv:begin_geo_latitude></evv:begin_geo_latitude> <!--Optional-->
               <evv:begin_geo_longitude></evv:begin_geo_longitude> <!--Optional-->
               <evv:begin_address1>1420 West 100 South</evv:begin_address1>
               <evv:begin_address2></evv:begin_address2> <!--Optional-->
               <evv:begin_city>Salt Lake City</evv:begin_city>
               <evv:begin_state>UT</evv:begin_state> <!--Optional-->
               <evv:begin_zip>84115</evv:begin_zip> <!--Optional-->
               <evv:end_geo_latitude></evv:end_geo_latitude> <!--Optional-->
               <evv:end_geo_longitude></evv:end_geo_longitude> <!--Optional-->
               <evv:end_address1>1420 West 100 South</evv:end_address1>
               <evv:end_address2></evv:end_address2> <!--Optional-->
               <evv:end_city>Salt Lake City</evv:end_city>
               <evv:end_state>UT</evv:end_state> <!--Optional-->
               <evv:end_zip>84115</evv:end_zip> <!--Optional-->
            </evv:main_record>
         </evv:EVV_Data>
      </evv:EVV_DataList>
   </soapenv:Body>
</soapenv:Envelope>
```

Appendix A

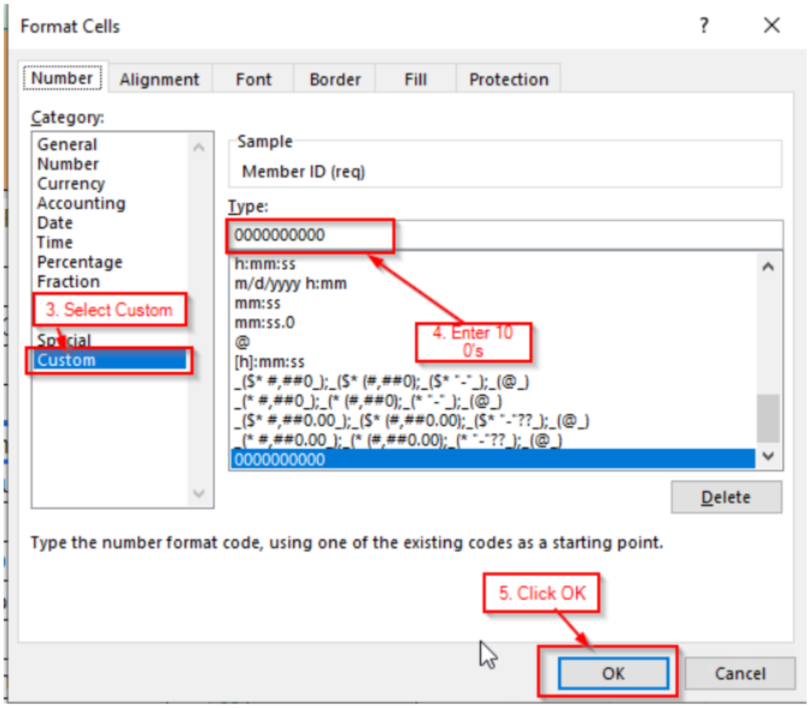
Since a CSV file cannot save custom formatting, if you open a CSV for any additional edits, any leading 0's (zeros) will be removed by Excel. If you open the CSV file in Excel, you will need to complete the steps below to prevent Excel from changing the Member ID column. **You will need to do this each time you open your CSV file in Excel.**

Figures 1 and 2: Steps for Keeping Member ID to 10 digits to address leading 0's



1. Right Click on Column header.

2. Select Format Cells



Format Cells

Number | Alignment | Font | Border | Fill | Protection

Category:

General
Number
Currency
Accounting
Date
Time
Percentage
Fraction
Special
Custom

Sample: Member ID (req)

Type: 0000000000

h:mm:ss
m/d/yyyy h:mm
mm:ss
mm:ss.0
@
[h]:mm:ss
_($* #,##0_);_($* (#,##0);_($* "-"_);_(@_)
_(* #,##0_);_(* (#,##0);_(* "-"_);_(@_)
_($* #,##0.00_);_($* (#,##0.00);_($* "-"??_);_(@_)
_(* #,##0.00_);_(* (#,##0.00);_(* "-"??_);_(@_)
0000000000

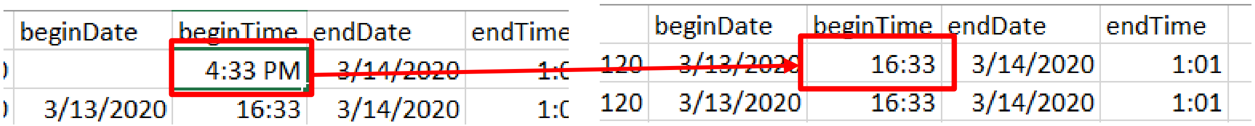
Type the number format code, using one of the existing codes as a starting point.

3. Select Custom

4. Enter 10 0's

5. Click OK

You will need to **add a space between the times in the Begin time and End time columns and the denotation of AM or PM**. Excel will convert it to a 24-hour clock format for you.



	beginDate	beginTime	endDate	endTime
)		4:33 PM	3/14/2020	1:(
)	3/13/2020	16:33	3/14/2020	1:(

	beginDate	beginTime	endDate	endTime
120	3/13/2020	16:33	3/14/2020	1:01
120	3/13/2020	16:33	3/14/2020	1:01

Click the **Choose File** button and select your file. Then click **Upload CSV**.